Mining Answers of On-Line Tutoring Tools

Brigitte Aguado, Agathe Merceron and Agnès Voisard

***Abstract*— With the emergence of e-learning, on-line tutoring tools are becoming more and more important. They change fundamentally the feedback that teachers can get from their teaching. What becomes now possible is to extract meaningful information from student homework available in e-form.**

The model presented in this paper serves as a support to represent, query, and mine student answers in this context. It provides extensive descriptive statistics of the classroom, including most common mistakes and shortest paths to solutions. It makes it possible to extract more sophisticated information. For instance, various classifications of students such as classification by mistakes, by mastered transformation rules, or by reasoning can be constructed, or one can look for mistakes often associated together. Our approach is illustrated with the Logic Tutor, developped at Sydney University.

I. Introduction

With the emergence of e-learning, flexible education, and the increasing number of students in some fields, on-line teaching tools are becoming more and more important. This changes fundamentally the nature of interaction between teachers and students. So far, managing student answers consisted in managing marks and computing statistics. What becomes now possible - and challenging - is the exploitation of student homework, which is available in electronic form.

Our work aims at analyzing and managing the answers obtained from students using on-line teaching tools. Relevant patterns can be conveyed in a meaningful way to both students and teachers. Students could find out their level, progress, and even compare those with respect to the rest of their group. Teachers, on the other hand, could examine this data under various angles to find trends, common mistakes, mistakes associated with each other, well-understood concepts, problems, progress, and so on, possibly grouping students by abilities, in order to readjust their teaching and provide proper feedback to their students. Several issues need to be addressed: managing students answers in a database, developing appropriate data mining techniques specific to the learning process, and visualizing and managing the results of the mining algorithms. As a first step, to validate our approach, we make use of an existing online teaching tool, the `Logic Tutor` currently in use at Sydney University. It is an on-line teaching assistant that allows students to practice formal proofs in propositional logic. This system is currently used in undergraduate teaching at the Information Technology School at Sydney University, in a course involving more than 400 students.

ESILV - Génie Informatique, Pôle Universitaire Léonard de Vinci, F-92916 Paris La Défense - Cedex (France) (email: brigitte.aguado@devinci.fr)

ESILV - Génie Informatique, Pôle Universitaire Léonard de Vinci, F-92916 Paris La Défense - Cedex (France) (email: agathe.merceron@devinci.fr)

Freie Universitaet Berlin Institut fuer Informatik Database and Information Systems Group Takustr. 9 D-14195 Berlin (email: voisard@inf.fu-berlin.de)

We aim at an approach that can be used for online courses in general, in particular for those offered by the emerging online campuses. Most presently-available tools provide help to display course material on-line, to facilitate electronic communication, to manage students marks, and so on. However, they do not include modules to get feedback from students on the concepts specific to a given course and to analyze their answers. The contribution of our project is to tackle this issue both theoretically and practically.

Querying and mining student answers require to model student's answer. Our approach is based on machine learning like techniques and bears similarities with ACM system [7] or production system [11] within the problem space paradigm [10]. Need to elaborate systems to classify students is well-known from traditional teaching. Various research work focus on the study of such systems (see [5], [4], [8]). For example, in [5] a system of classifying student errors from essay exam answers is developped in the context of an introductory microeconomics course. These works are not linked to any on-line teaching tool. Some tutoring systems [3] implement human teaching strategies and tactics determining for example task difficulty and degree of assistance. Relevant classifications of student answers may contribute to improve these strategies.

This paper is organized as follows. Section 2 describes the model that serves as a support for handling student answers as well as the basic queries that one would like to pose against the proposed structure. Section 3 describes more elaborate operations and introduces techniques from data mining. Section 4 illustrates our approach in the context of the `Logic Tutor` mentioned previously. Finally, Section 5 draws our conclusions.

II. Model

This section first introduces the structure that represents the reasoning of the students. It includes both the valid and the possible invalid reasoning. Queries can be posed against this structure to extract meaningful information. They are informally presented in the second part of this section.

Before we proceed, we find it useful to roughly clarify the kind of on-line teaching tools we have in mind. We think of tools that propose problems or exercises to students on a specific topic, like the `Logic Tutor` [1] proposes logic exercises. Solving a problem may necessitate a number of intermediary steps. At each step, the student gives some intermediary result together with a justification. Either the result and justification are correct or there is a mistake somewhere. In the latter case, the tutor gives an error message containing some explanation to the student who, then, may re-do the step. The exercise finishes when the student either reaches the solution, or gives up.

A. Problem solving structure (PSS)

We need to consider a collection of exercises that will be chosen and solved by groups of students. Many students may choose the same exercise. An exercise has a certain difficulty. It has a finite number of solutions and is solved by each student in a sequence of steps. Each step of the reasoning is based on a justification. It is valid or invalid. In case it is invalid, the system - the tutor - notifies the student who makes another try either with another result, or with another justification, or both. Note that the notion of justification is context dependent. For instance, in the case of the `Logic Tutor`, a justification is a rule applied to formulas. An invalid step leads to backtracking in the reasoning. A correct answer is a list of valid steps leading to a solution.

The most natural structure that comes to mind to model the problem solving evolution is a tree associated with a student and an exercise. At each level, a step in the reasoning is represented, which is the justification, or rule, used for the edge, and the intermediate result for the node. An invalid justification will lead to an invalid node, which is a leaf of the tree - or, in other words, a dead end in the reasoning. If the justification and the result are correct, the justification leads to a valid node and the process continues until a solution is reached. This is similar to workflows. The root of the tree is the problem to solve.

Definition 1
A solved exercise (SE) is a 4-tuple:

$$SE = (St(l), l_n, PS, PSS),$$

where $St(l)$ is a student with a certain level, l_n is a difficulty (which can be defined with respect to the level of a student as an exercise can be easy for a "good" student and hard for a mediocre one). The difficulty associated with an exercise is an integer between 1 and n, with n being a predefined parameter. Finally, PS is the problem to solve and PSS is a problem solving structure as defined below.

Definition 2
A problem solving structure (PSS) is defined as a tree:

$$PSS = (N, E),$$

where N is a finite set of *nodes* such that $PS \in N$, $E \subseteq N \times N$ is a finite set of *edges*. PS is the node that forms the root of the PSS, i.e., it is the problem statement.

More precisely,

- $N = N_{val} \cup N_{inval}$ is the set of all nodes, where N_{val} represents the set of valid nodes and N_{inval} the set of invalid nodes.
- $E = E_{valid} \cup E_{invalid}$ is the set of edges. The set E_{valid} is formed with the correct justifications provided by the student while the set $E_{invalid}$ is formed with the error messages given by the system.

There is a valid edge e from $n_i \in N$ to $n_{i+1} \in N$ if n_i is the step proposed by a student as a valid partial solution to the problem, namely, if a valid justification was used and a correct result given. There is an invalid edge between $n_j \in N$ and $n_{j+1} \in N$ if an invalid justification was proposed or mistake has been done in the result, or both. This leads to an invalid node, i.e. $n_{j+1} \in N_{inval}$.

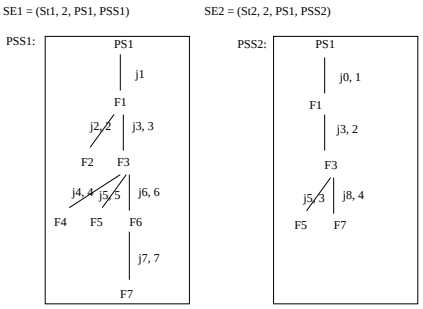

Fig. 1. Problem solving structures.

Edge labeling.
An edge is labelled with (i) the justification used at this step or the error message generated at this step and (ii) the order of the justification in the reasoning at that stage (remember that students may try a solution and backtrack).

Node labeling.
A node encompasses the valid or invalid result in the deduction. The root of the tree contains the problem statement.

Solution.
A correct answer is a path of the PSS leading to a solution leaf. An *incomplete answer* is a PSS whose leaves are either invalid nodes or nodes that are different from the solution.

These concepts are summed up in Figure1.

Figure1 represents two solved exercises, SE1 and SE2 with the same problem statement and the same level of difficulty with respect to the individual levels of the students. With each SE is associated a PSS, namely PSS1 and PSS2. Invalid edges are represented as thick lines. As we can see from the left hand side of the figure (PSS1), justification j2, the first one following F1 cannot be applied starting F1 and student St1 needs to backtrack in her/his reasoning and try another justification. The student choses justification j3 as a second try, which leads to F3. Note that many mistakes can be made at a certain step, e.g., j4 and j5 at Level 3 in PSS1. Finally, the fourth step in PSS1 represents the end of the problem solving - the only valid leaf, F7. Note that in PSS2 the solution was found earlier by St2. Note also the importance of ordering edges as illustrated in PSS1 between levels 3 and 4, for instance. This will help us to do some reasoning on the reasoning.

B. Querying the structure

Useful information for self-evaluation and class evaluation can be obtained by performing tree searches either on individual PSS's or on forests of PSS's. They are described below in the form of queries that can be posed against the structure described above.

1) Queries on individual students: Such queries allow one to check on the progress of a particular student, for instance, by comparing the number of invalid nodes in different SE's for the same student and at the same level of difficulty.

2) Queries on groups of students: These queries allow one to compare the results of a class and to group students according to certain criteria. The number of useful queries in this context is obviously large. Besides, they may concern a single exercise or a set of exercises. Such queries include:

- Who are the students who did not find any solution to exercises PS1, PS2, and PS3?
- Who are the students who found the optimal solution to exercise PS1?
- Who are the students who always tried invalid rules at each step of exercise PS1?
- Who are the students who have exactly the same solutions?

The results of the queries can be obtained directly from querying the reference structures. Moreover, basic statistical operations can be performed. More elaborate information, such as correlations between results or pattern discovery, can be extracted through data mining techniques as described in the following section.

III. Further Operations

In this section, we introduce operations to allow the teacher to gain a finer insight of the classroom, and a finer insight of the mistakes. Hierachical classifications serve the former goal, associations the latter. Both techniques are well known in the data mining field [6]. We adapt them to the tutoring context.

A. Hierachical classifications

The goal of a hierachical classification is to group individuals in homogeneous classes, where homogeneity is measured by means of a distance between individuals and groups. The number of classes is not known a priori. Rather, classification is stopped when the distance between two groupes is too big and does not garantee anymore the homogeneity of the individuals grouped together.

General algorithm.
We suppose that we have a population of N individuals and m attributes. Each individual is characterized by the values taken for each of the attributes. This general algorithm makes use of two different distances, an *initial distance*, which measures how different two individuals are, and a *distance between groups*, and we will describe these distances shortly. The algorithm finds the classes as follows.

1) Choose a distance between two individuals.
2) Pairwise, calculate all initial distances between individuals. This gives a $N \times N$ triangular matrix of distances, triangular because a distance is symmetric. At this stage, the N individuals are seen as N classes, each class being composed of 1 individual only.
3) Select two classes with the shortest distance between them.
4) Group these two classes into a new one. Thus, the number of classes diminishes by 1.
5) Update all distances between already existing classes and the one newly formed.
6) Repeat step 3 to step 5 till the shortest distance above a given threshold is reached or till there is only one class.

	1	2	3	4	5
1	0				
2	1	0			
3	5.0	4.5	0		
4	8.5	7.8	3.6	0	
5	7.2	6.7	2.2	2	0

Fig. 2. Matrix of initial distances.

	1, 2	3	4	5
1, 2	0			
3	4.5	0		
4	7.8	3.6	0	
5	6.7	2.2	2	0

Fig. 3. Matrix of distances after the first iteration of the algorithm.

The *initial distance* is used in step 2 to get initial distances. Its choice depends on the particular data, individuals and attributes that the algorithm has to classify. Its choice is crucial as it determines what aspects of individuals the classification concentrates on, and below we show how we proceed in our context. The *distance between groups* is used repeatedly in step 5. We introduce three commonly used methods to define a *distance between groups*. With the single linkage method, the distance between two groups is given by the minimal distance separating two individuals of each group. With complete linkage, it is the maximal distance between two individuals of each group that gives the group distance. With the average method, the distance between two groups is calculated summing all possible distances between any two individuals of each group and dividing this sum by the number of possibilities, which is $|\, g_1 \,| \times |\, g_2 \,|$. Formally:

Let g_1, g_2 be two groups and $d_i(x, y)$ denote the initial distance between individuals x and y.

- *Single linkage*:
  $d(g_1, g_2) = min_{x \in g_1, y \in g_2} d_i(x, y)$.
- *Complete linkage*:
  $d(g_1, g_2) = max_{x \in g_1, y \in g_2} d_i(x, y)$.
- *Average*:
  $d(g_1, g_2) = \frac{\Sigma_{x \in g_1, y \in g_2} d_i(x, y)}{|g_1| \times |g_2|}$.

As an illustration for the whole algorithm, consider 5 individuals and suppose that an initial distance gives the 5×5 matrix of step 2 as shown in Figure 2. Suppose that we choose *single linkage* to generate the group distance. The two classes with the minimal distance are $\{1\}$ and $\{2\}$, so they are grouped into a new one, $\{1, 2\}$. Completion of step 5 gives the matrix shown in Figure 3. The distance between $\{3\}$ and $\{1\}$ is 5 while distance between $\{3\}$ and $\{2\}$ is 4.5. Using single linkage, the minimum is taken for the distance between $\{3\}$ and $\{1, 2\}$. A similar procedure is followed to obtain the distance between $\{4\}$ and $\{1, 2\}$, and between $\{5\}$ and $\{1, 2\}$. The minimum distance of the whole matrix is now between the classes $\{4\}$ and $\{5\}$, which gives the new class $\{4, 5\}$ After, the second, then the third iteration we obtain the matrices shown in Figure 4. Hierarchical classification is summarized graphically in the form of a dendrogram. The dendrogram obtained in this example is shown Figure 5.

	1, 2	3	4, 5
1, 2	0		
3	4.5	0	
4, 5	6.7	2.2	0

	1, 2	3, 4, 5
1, 2	0	
3, 4, 5	4.5	0

Fig. 4. Matrix of distances after the second iteration of the algorithm.

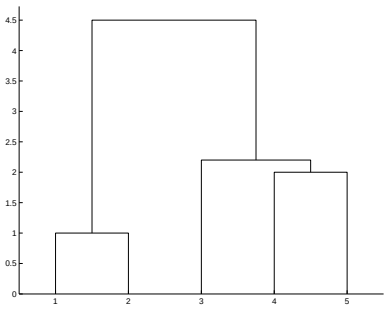

Fig. 5. The dendrogram obtained for the example Figure 2.

The general algorithm gives different classifications, depending on the initial distance chosen (and, to some extent, on the group distance, though the impact of the group distance is quite well understood [6]).

We propose different ways of associating attributes to problem solving structures, and different choices of initial distances, each way gives a specific insight into the classroom.

Insight into mistakes.
To classify students by mistakes is to group together students making the same mistakes. A student is characterized by the answer he gave to a particular exercise. Let $SE_{i,1\leq i\leq N}$, be the N structures obtained from N students to a given exercise PS. First, we construct the set M_E of all error messages occurring in the N structures. Thus, $M_E = \{m_1, \ldots, m_M\}$ if they are M different error messages occuring. With this set, we build a table with N lines for the N structures SE_i and M columns with a 1 in line (i, j) if structure SE_i contains an invalid edge with label m_j, and 0 otherwise, as shown in Figure 6. In other words, (i, j) is 1 if and only if student i has made mistake with error message m_j while solving exercise PS.

We obtain a table having only binary attributes. Focusing on mistakes, absence of mistakes is not informative. In other words, it makes no sense to take into account in the distance between i and j attributes where both i and j get 0. However, it makes sense to take into account attributes where both i and j get 1, as they both make the same mistake and may both need further explanations from the teacher for the same topic. Therefore, we propose two distances. The first one is defined focusing on non-common mistakes, while the second one (based on Dice coefficient) is defined focusing on mistakes that both students have made. Let q be the number of attributes where both i and j get 1, r be the number of attributes where i gets 1 and j gets 0, and t be the number of attributes where i gets 0 and j gets 1.

	m_1	m_2	m_3
1	1	1	1
2	1	1	0
3	0	0	1
4	0	0	1

Fig. 6. Table of mistakes for 4 students and 3 error messages.

1) $d_N(i,j) = \frac{r+t}{q+r+t}$.
2) $d_D(i,j) = 1 - \frac{2q}{2q+r+t}$.

Taking the individuals of Table 6, one has:
$d_N(1,2) = \frac{1}{3} = 0.33$, $d_N(1,3) = d_N(1,4) = \frac{2}{3} = 0.66$,
$d_N(2,3) = d_N(2,4) = \frac{3}{3} = 1$, $d_N(3,4) = \frac{0}{1} = 0$.
$d_D(1,2) = 1 - \frac{4}{5} = 0.2$, $d_D(1,3) = d_D(1,4) = 1 - \frac{2}{4} = 0.5$,
$d_D(2,3) = d_D(2,4) = 1 - \frac{0}{3} = 1$, $d_D(3,4) = 1 - \frac{1}{1} = 0$.

This can be easily extended in the case where answers to several exercises, instead of one, are taken into account (the exercises need not to be the same for all students). The set M_E is simply error messages from all invalid edges seen in all structures.

Insight into mastered rules.
To obtain a classification where students grouped together master similar know-how is similar to the classification by mistakes. The difference lies in the column of the table. To build the table, one takes the set J containing all correct justifications of all valid edges contained in the N structures.

Insight into mistakes and mastered rules.
To obtain a classification where students grouped together make similar mistakes and master similar knowledge is also done in a similar way. The columns of the table are obtained taking $M_E \cup J$.

Insight into reasoning.
To classify students by reasoning is to group together students who obtain a solution in the same straightforward way. For example, students who have found a shortest solution straightway should go in the same class. Students who first try one alternative before they engage on the way to the shortest path to a solution should belong to other classes, depending on how long they went in the wrong direction. To achieve a sensible classification, we do not apply hierarchical classification starting with the whole population, rather we perform an initial clustering and apply hierarchical classification on each of these initial clusters.

Let $\pi_1, \ldots, \pi_n$ the n different paths to solutions for the exercise that has been solved. These paths are grouped by length, which gives $l_1, \ldots, l_m$, $m \leq n$. l_1 contains all paths to a solution with the shortest path length, l_2 contains all paths to a solution with the second shortest path length, and so on. We make an initial clustering of structures. This initial clustering contains at most $m + 1$ clusters, $C_0, C_1, \ldots, C_m$. C_0 is the set of structures not containing a solution path, C_1 is the set of structures containing a solution from set $l_1, \ldots, C_m$ is the set of structures containing a solution from set l_m.

We perform a hierarchical classification on each individual cluster C_i. First, the set J_O of all justifications of all valid edges occurring in trees from C_i outside the path to the solution is constructed. J_O contains all justifications that did not contribute to the solution. With this set, one builds a table with $\mid C_i \mid$ lines ($\mid C_i \mid$ means cardinality of C_i) and $\mid J_O \mid$ columns

	j_1	j_2	j_3
1	0	0	0
2	1	1	0
3	0	0	2

Fig. 7. Table of justifications not leading to the solution.

with a in line (i, j) if tree PSS_i contains a occurrences of valid egdes with justification j_j outside the path to solution. Such a table is shown in Figure 7. In other words, (i, j) is a if and only if student i has used a times justification j_j for the wrong purpose, because it did not help to find the solution.

We have a table with quantitative attributes. Among existing distances on quantitative attributes, those that are relevant in our context are those that take into account all attributes and that do not standardize values. The *generalised euclidean distance* is the one from geometry generalized to an arbitrary number of attributes. If some justifications are more important than others, the teacher might want to use weights as in the *weighted generalised euclidean distance.* The *manhattan distance* simply adds the differences (in absolute value) obtained for each attribute.

1) *Generalised euclidean distance:*
   $d_e(i, j) = \sqrt{\Sigma_{k=1}^{k=|J_O|}((i, k) - (j, k))^2}$.
2) *Weighted generalised euclidean distance:*
   $d_w(i, j) = \sqrt{\Sigma_{k=1}^{k=|J_O|} w_k \times ((i, k) - (j, k))^2}$, where w_k is a weigth.
3) *Manhattan distance:*
   $d_m(i, j) = \Sigma_{k=1}^{k=|J_O|} \mid ((i, k) - (j, k)) \mid$.

Taking the individuals of Table 7, one has:
$d_e(1, 2) = \sqrt{1 + 1} = 1.41$, $d_e(1, 3) = \sqrt{4} = 2$,
$d_e(2, 3) = \sqrt{1 + 1 + 4} = 2.45$.
$d_m(1, 2) = 1 + 1 = 2$, $d_m(1, 3) = 2$,
$d_m(2, 3) = 1 + 1 + 2 = 4$.

Contrary to the mistakes or mastered rules case, it is not obvious how to extend this classfication taking several exercises into account.

B. Associations

Quite often students make more than one kind of mistakes. If a teacher is aware of mistakes that often occur together while solving an exercise, she may take this fact into account in her teaching. The goal of association mining techniques is to find items, in our case mistakes, often occuring together.

General Algorithm.
We suppose that we have a population of N individuals and each individual is characterized by a list of items. Figure 8 gives an illustration. Individual 1, for instance, is characterized by the list $(1, 2)$.

Items often occuring together are given by rules of the following form:
$3 \rightarrow 4$, *support* 0.4, *confidence* 0.66, or
$1 \rightarrow 2$, *support* 0.2, *confidence* 0.66.
The first rule means that if item 3 is present, then item 4 is also present. This is supported by 40% of the individuals with a confidence of 60% .

	Item list
1	(1, 2)
2	(1, 2, 3)
3	(1, 3, 4)
4	(2, 4)
5	(2, 4, 5)
6	(2, 5)
7	(3, 4)
8	(3, 4, 5)
9	(3, 4, 5, 7)
10	(3, 5, 6)

Fig. 8. 10 individuals and their list of items.

The concepts *support* and *confidence* have a precise meaning that we introduce now. Let $t_{i,1\leq i\leq N}$, be N lists of data, and I be the sets of items occurring in all $t_{i,1\leq i\leq N}$. In our example, we have $I = \{1, 2, 3, 4, 5, 6, 7\}$. One is looking for rules of the form $X \rightarrow Y$, with $X, Y \subseteq I$ having *support* and *confidence* above a minimum threshold.

- Support: $sup(X \rightarrow Y) = \frac{|\{t_i | X \cup Y \subseteq t_i\}|}{N}$.
- Confidence: $conf(X \rightarrow Y) = \frac{|\{t_i | X \cup Y \subseteq t_i\}|}{|\{t_i | X \subseteq t_i\}|}$.

The concept of support is to make sure that only items occurring often enough in the data will be taken into account to establish the association rules. Confidence measures whether Y is really implied by X. If X occurs a lot anyway, then almost any subset Y could be associated with it. A confidence which is high enough makes sure that X and Y have some causal link.

The algorithm [2] works by constructing several extra lists. First, list l_1 of single mistakes having the desired support is constructed. Let us take a minimum support 20% with the data of Figure 8. $sup(1) = \frac{3}{10} = 30\%$, hence $sup(1) > 20\%$, so item 1 belongs to l_1. Making similar calculations for all items leads to $l_1 = (1, 2, 3, 4, 5)$. From l_1, one deduces the list l_2 of pairs having a support above or equal to the minimum. For example $sup(1, 2) = \frac{2}{10} = 20\%$, but $sup(1, 4) = \frac{1}{10} = 10\%$. Making similar calculations for all possible pairs of the list l_1 gives in our example $l_2 = ((1, 2), (1, 3), (2, 4), (2, 5), (3, 4), (3, 5), (4, 5))$. Then, the list l_3 of triples with support above or equal to the minimum is constructed, which gives here $l_3 = ((3, 4, 5))$. In our example no extra list can be added because no quadruple has enough support. From each list $l_i, i > 1$, one tries any combination and only rules with a confidence above the desired threshold are kept. For example, $conf(2 \rightarrow 4) = \frac{2}{3} = 66\%$, or $conf(2 \rightarrow 4) = \frac{2}{5} = 40\%$. Taking rules having a confidence greater or equal to 60%, we get the association rules shown in Figure 9.

Associations for mistakes.
To adapt the general algorithm to our context is straightforward. Let PSS_i be the tree representing the answer of student i. From this tree we construct the list t_i which contains all error messages of all invalid edges of tree PSS_i, i.e., all mistakes made by this student. The set of items I is then formed by all error messages of invalid edges occurring in the N trees. The data

Rule	Support	Confidence
$1 \rightarrow 2$	20%	66%
$1 \rightarrow 3$	20%	66%
$3 \rightarrow 4$	40%	66%
$4 \rightarrow 3$	40%	66%
$5 \rightarrow 3$	30%	60%
$5 \rightarrow 4$	30%	60%
$4, 5 \rightarrow 3$	20%	66%
$3, 5 \rightarrow 4$	20%	66%

Fig. 9. Association rules obtained from the data Figure 8.

obtained that way can be mined for associations using the general algorithm.

In a teaching context, it makes sense to have a support and a confidence which are not too low. Indeed, only mistakes occuring often enough together need special care in teaching. Mistakes that occur seldom are given by the descriptive statistics and may need some specific action with the concerned students, not necessarily a revision of the teaching material like mistakes associated together might need.

Associations for mistakes can be easily extended in the case where answers to several exercises, instead of one exercise, are taken into account (the exercises need not to be the same for all students). An algorithmic issue worthwhile to explore is to deduce associations for a set of exercises from the associations obtained for each exercise separately.

IV. Illustration

The main motivation for our approach comes from the `Logic Tutor` [1], an e-tool to train students in a special field of logic called the logical proofs. It has been used for two years now in the course 'Languages and Logic' followed by more than 400 students enrolled in computer science at the University of Sydney. The impressive amount of data collected led us to the idea of exploiting them in a way that was never done, but also not possible, before.

A. General presentation of the `Logic Tutor`

An exercise is a set of formulas from propositional logic. This set is composed of the premisses plus one particular formula called the conclusion. To solve an exercise is to derive the conclusion from the premisses by applying rules of logic. A step of the derivation works as follows. The student selects formulas, either premisses or formulas already derived, applies a logical rule to them and obtain a new formula which is added to the set. If the student makes a mistake in a step, the tutor gives immediate feedback about the nature of the mistake and a tip to correct it. The derivation stops when either the last formula obtained by the student is equal to the conclusion, or the student gives up.

As an example consider the exercise below with 4 premisses and the conclusion introduced by $\vdash$.

1- $(A \vee (B \rightarrow D))$
2- $(\neg C \rightarrow (D \rightarrow E))$
3- $(A \rightarrow C)$
4- $\neg C$

$Prem.$	$No.$	$Form.$	$Just.$	$Refs.$
3	1	$(A \rightarrow C)$	P	
4	2	$\neg C$	P	
3, 4	3	$\neg A$	MT	1, 2

Fig. 10. A step of the logic exercise.

$Prem.$	$No.$	$Form.$	$Just.$	$Refs.$
3	1	$(A \rightarrow C)$	P	
4	2	$\neg C$	P	
3, 4	3	$\neg A$	MT	1, 2
3, 4	4	$\neg A \vee B$	$Add.$	3
1	5	$(A \vee (B \rightarrow D))$	P	
1, 3, 4	6	$(B \rightarrow D)$	DS	3, 5
2	7	$(\neg C \rightarrow (D \rightarrow E))$	P	
2, 4	8	$(D \rightarrow E)$	MP	2, 7
1, 2, 3, 4	9	$(B \rightarrow E)$	HS	6, 8

Fig. 11. A possible answer.

$\vdash (B \rightarrow E)$.

Three steps of a derivation are shown in Figure 10.

The column $Prem.$ refers to the premisses used, $No.$ to the line number of the derivation, the column $Form.$ is a formula, $Just.$ is the logical rule used and $Refs.$ are the lines the logical rule applies to. The special logical rule that allows to write line 1 and 2 is P indicating that these two formulas are premisses. The logical rule that allows to write formula $\neg A$, line 3, is MT for *Modus Tollens* and it applies to lines 1 and 2, making use of premisses 3 and 4.

Suppose that the student makes a mistake, and uses the justification MP for *Modus Ponens* instead of MT. The `Logic Tutor` checks step by step the answer of the student. This step is checked as incorrect because *Modus Ponens* cannot be applied to the two lines entered by the student. The `logic tutor` looks for a reason to the mistake. The first search is to check whether the student made a mistake in the choice of the rule, i.e., whether another rule applies to the two lines indicated by the student. This is the case here. The `Logic Tutor` rejects the input of the student and produces the following error message *invalid justification Modus Ponens cannot be applied, try Modus Tollens instead*. The input of the student is rejected, so the student has to give another input, however the mistake is saved by the tool under the name *invalid justification Modus Ponens*.

A possible complete answer to this exercise like it would be in the final window on the screen is shown in Figure 11. As already mentionned, P means *premisse*, MT means *Modus Tollens*, Add stands for *Addition*, DS stands for *Disjunctive Syllogism*, MP means *Modus Ponens*, and HS stands for *Hypothetical Syllogism*. The mistake does not show up because the `Logic Tutor` does not accept wrong inputs. Rather, it helps students to correct their mistakes and to produce correct inputs.

B. Casting answers into problem solving structures

Presently, complete answers, including mistakes, are stored and part of the extensive descriptive statistics as described ear-

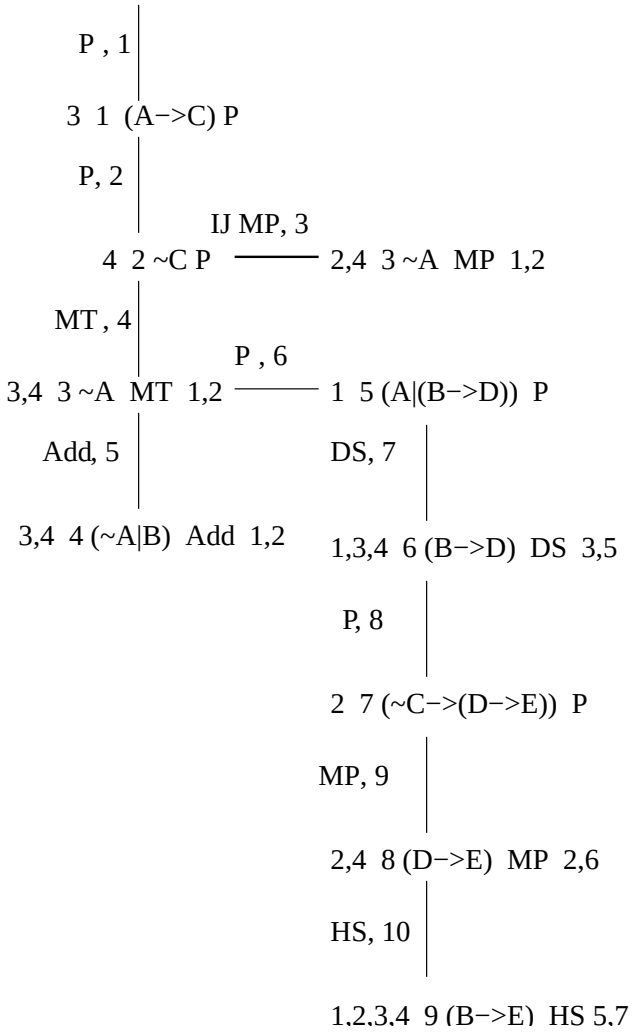

Fig. 12. A problem solving structure for the logic exercise.

lier in the paper can already be obtained. Statistics on mistakes have already proven useful for revisions [9].

We illustrate on our running example how an answer is cast into a tree structure as described in Section II. The resulting tree for the answer of Fig. 11 is shown in Fig. 12. Let us call our exercise, composed of the four premisses and the conclusion, PS. The root is labeled with PS. Each node contains an input as given by the student, while each edge gives either the rule used in the input, or the error message given to the student by the system.

One notices that the present tree has 3 leaves, one is a invalid node because of the mistake. The node using the rule *Addition* shows a wrong direction taken by the student. The last leaf is the solution.

V. Conclusion

In this paper, we have presented a general model to represent student answers obtained from on-line tutoring tools. Our approach uses a tree structure capturing both mistakes and correct reasoning of students. This model allows to process exercises more general than multiple choice ones.

Thanks to this support, meaningful information from a pedagogical viewpoint can be delivered to teachers who can then personalize and adjust their teaching. This includes classification of students by mistakes, by mastered rules or by reasoning. Furthermore, we mine mistakes for associations. This approach of finding mistakes often occurring together contributes to a new feedback for teachers who may take it into account in the elaboration of their course material.

We have illustrated our model taking the `Logic Tutor`, an e-tool to train students in formal proofs and currently in use at the University of Sydney.

Our current works follows two directions. First, we are implementing the association algorithm to mine the database of the `Logic Tutor`. Indeed, finding patterns of mistakes often occurring together could lead to revise the way logical rules are introduced to students in the logic course where the tool is used.

Second, we are validating our approach with further exercises that fall in the scope of on-line teaching tools we have in mind. As a first example, we took an exercise on *if statement* solved by undergraduate students following a course in "Introductory Programming" at the University Leonard de Vinci in Paris. We have performed a hierarchical classification by mistakes of the students following the method presented in Section III taking distance d_N. The result shows a strong correlation between the classification by mistakes we have obtained and the ranking of the students obtained by traditional way (human marks), which is encouraging. This preliminary work has put in evidence a short-coming of currently available tools offering hierarchical classification, especially the drawing of dendrograms. As a future work, we want to improve the readability of dendrograms for teachers and obtain an informative sequence of the students along the horizontal axis.

Future research includes also an extension of insight into mistakes or into mastered rules where not only mistakes or mastered rules are taken into account, but also their order. For example if a student makes a mistake, notices it and correct it by applying a correct transformation rule associated to the mistake, a teacher might consider that the student has understood the mistake and does need further explanations. Several variations are possible and need to be explored.

Finally, future research includes exploring other data mining algorithms and their application to the education context as well as the study of a general platform to ease the development of on-line tutoring tools.

References

[1] D. Abraham, L. Crawford, L. Lesta, A. Merceron and K. Yacef, "The Logic Tutor: A Multimedia Presentation',' *Interactive Multimedia Electronic Journal of Computer-Enhanced learning*, Vol. 3, Nb. 2, Nov. 2001

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," in *Proceedings of the 20th VLDB Conference*, Santiago, Chile 1994

[3] B. du Boulay, R. Luckin, "Modeling Human Teaching Tactics and Strategies for Tutoring Systems," *International Journal of Artificial Intelligence in Education*, Vol. 12, pp. 235-256, 2001.

[4] M. Druger, "A Perspective on Exams and Grading", *Journal of College Science and Teaching"*, Vol. 30, Issue 3, pp. 210-211, 2000.

[5] C. Ellard, M. Feinberg, J. S. Siekpe, "Classifying student errors in the introduction to microeconomics course," *Business Quest, Journal of Applied Topics in Business and Economics*, 2002.

[6] J. W. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[7] P. Langley, S. Ohlsson, "Automated Cognitive Modeling," *Proceedings of the Second National Conference on Artificial Intelligence*, 1984.

[8] N. Lee, “Notions of Errors and Appropriate Corrective Treatment,” *Hong Kong Papers in Linguistics and Language Teaching*, Vol. 14, pp. 55-70, 1990.

[9] L. Lesta and K. Yacef, “An Intelligent Teaching-Assistant System for Logic,” *Proceedings of Intelligent Tutoring Systems*, Biarritz, France, Springer-Verlag, June 2002.

[10] A. Newell, “Reasoning, Problem Solving, and Decision Processes: The Problem Space Hypothesis,” in R. Nickerson (Ed.), *Attention and Performance*, Hillsdale, Lawrence Erlbaum Associates, 1980.

[11] A. Newell, H. Simon, “Human Problem Solving,” Englewood Cliffs, Prentice-Hall, 1972.

[12] R. Sison, M. Shimura, “Student Modeling and Machine Learning,” *International Journal of Artificial Intelligence in Education*, Vol. 9, pp. 128-158, 1998.